A Clustering Algorithm for Intrusion Detection

Qiang Wang Vasileios Megalooikonomou
Data Engineering Laboratory (DEnLab)
Department of Computer and Information Sciences
Temple University
1805 N. Broad Street, Philadelphia, PA 19122, USA

ABSTRACT

In this paper, we introduce a new clustering algorithm, *FCC*, for intrusion detection based on the concept of fuzzy connectedness. This concept was introduced by Rosenfeld in 1979 and used with success in image segmentation; here we extend this approach to clustering and demonstrate its effectiveness in intrusion detection. Starting with a single or a few seed points in each cluster, all the data points are dynamically assigned to the cluster that has the highest fuzzy connectedness value (strongest connection). With an efficient heuristic algorithm, the time complexity of the clustering process is $O(N\log N)$, where N is the number of data points. The value of fuzzy connectedness is calculated using both the Euclidean distance and the statistical properties of clusters. This unsupervised learning method allows the discovery of clusters of any shape. Application of the method in intrusion detection demonstrates that it can detect not only known intrusion types, but also their variants. Experimental results on the KDD-99 intrusion detection data set show the efficiency and accuracy of this method. A detection rate above 94% and a false alarm rate below 4% are achieved, outperforming major competitors by at least 5%.

Keywords: intrusion detection, data mining, clustering, fuzzy connectedness

1. INTRODUCTION

Today, with more and more computers getting connected to public accessible networks (e.g., the Internet), it is impossible for any computer system to be claimed immune to network intrusions. Since there is no perfect solution to prevent intrusions from happening, it is very important to be able to detect them at the first moment of occurrence and take actions to minimize the possible damage.

As defined in [1], intrusion detection is “the process of monitoring the events occurring in a computer system or network and analyzing them for signs of intrusions, defined as attempts to compromise the confidentiality, integrity, availability, or to bypass the security mechanisms of a computer or network”. Before data mining techniques are introduced into this field, intrusion detection was heavily dependent on a manually maintained knowledge base which contained signatures of all known attacks. Features of monitored network traffic were extracted and then compared with these attack signatures. Whenever a match was found, an intrusion was claimed to be detected and it was reported to the system administrator. Due to the difficulty and expense to manually maintain the knowledge base to reflect the ever-changing situations, it was not feasible to continue working in this traditional way. Recently, many researchers turned into data mining techniques to attack the problem [2, 5, 7, 9, 11].

Data mining based intrusion detection systems can be roughly categorized into two major groups: misuse detection and anomaly detection. In misuse detection, a model is trained with labeled data to recognize the patterns of “normal” visits and “intrusion” attempts. Unlike the traditional knowledge base method, signatures of different types of intrusions are learnt automatically, and they are much more powerful than manually defined signatures in recording the subtle characteristics. Misuse detection has been shown to be very successful in detecting previously known attacks. However, since the misuse model is highly dependent on the labeled data used in the training stage, its capabilities of detecting new intrusion types is limited. Different from misuse detection, anomaly detection first establishes a model of normal system behaviors, and anomaly events are then distinguished based on this model. The implicit assumption is that any intrusive activity will be anomalous. Anomaly detection is able to detect newly emerging attacks (if only the assumption still holds), but it also has some drawbacks. It may fail to detect some known attacks if the behaviors of them are not significantly different from what is considered to be normal. Moreover, the false alarm rate tends to be high when the data of some normal system behaviors are not involved in the training phase.

Phung [6] listed several major shortfalls with traditional intrusion detection systems. With the help of data mining approaches, these difficulties can be easily overcome. Briefly, the data mining techniques have provided the following benefits:

- *Improved variants detection*. This is especially true for anomaly detection. Not limited to pre-defined signatures, the concern with variants is not as much as before, since any deviation from a unique (normal) signature will be treated as intrusion, including those previously unknown variants of intrusions.
- *Controlled false alarms*. Even though these are false positives, with a learning process to identify recurring sequences of false alarms, it is possible for us to filter those normal system activities and keep the rate of false alarms at an acceptable level.
- *Reduced false dismissals.* With data mining techniques, patterns (or signatures) of normal activities and abnormal events (intrusions) can be created automatically. It is also possible to introduce new types of attacks through an incremental learning process. As a result, more and more attacks can be detected correctly. This leads to a reduced number of false dismissals.
- *Improved efficiency*. One very attractive feature of data mining techniques is the ability to extract most meaningful information out of large amounts of data. After a step of feature extraction and/or feature selection, the learning process can be done much more efficiently.

A wide variety of data mining techniques have been applied to intrusion detection, which include decision trees [5] and neural networks [9]. With pre-labeled (normal or intrusion) network traffic data, classification-based intrusion detection models can be trained and proved to be very useful. In real life, however, it is not always possible to have enough training data to guarantee the effectiveness of the classifiers. At the same time, temporal changes in network intrusion patterns and characteristics also tend to invalidate the usability of trained models. These challenges lead to the emerging interests of researchers in applying unsupervised or clustering techniques to intrusion detection.

Recently, several unsupervised methods have been proposed. Portnoy et al. [7] introduced an algorithm to detect both known and new intrusion types without the need to label the training data. This method, however, has its limitations. It is based on the assumption that “the normal instances constitute an overwhelmingly large portion (>98%)”. Also, it requires a predefined parameter of clustering width which is not always easy to find. Y-means, introduced by Guan et al. [2], overcomes the shortcomings of the traditional K-means algorithm and can automatically decide the number of clusters through splitting and merging clusters. However, like other centroid-based clustering algorithms, Y-means can only deal with clusters with a spherical shape and one still needs to define a threshold for the "confident area".

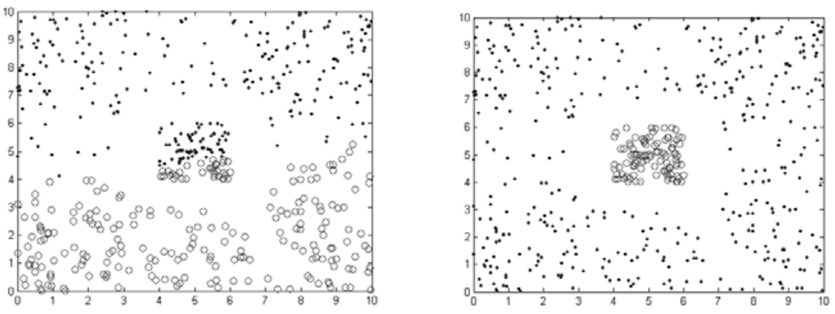
To deal with these issues, we introduce a new clustering algorithm, the Fuzzy-Connectedness Clustering (*FCC*) for intrusion detection based on the concept of fuzzy connectedness. With little prior knowledge, the proposed method allows the discovery of clusters of any shape and can detect not only the known intrusion types, but also their variants.

The rest of the paper is organized as follows. Some background information about fuzzy connectedness is provided in the next section. We then present the proposed clustering algorithm: *FCC*, followed by experimental results. Finally, we present concluding remarks and suggestions for future study.

2. BACKGROUND

In data mining, clustering is the most important unsupervised learning process used to find the structures or patterns in a collection of unlabeled data. During the learning process, the similarities between pairs of data instances are considered and an optimized output is found which maximizes the similarities within the same group and the dissimilarities between different groups. While there is no absolute criterion to judge the optimality of the clustering result, the choice of similarity measures can surely lead to totally different groupings.

In many clustering algorithms, the spatial distance (usually Euclidean distance) between different instances is used to measure the similarity. This has proved to be successful in many applications. It is intuitive that similar objects are closer together while objects from different groups are far from each other. This intuition, however, is not always true in more complicated applications, especially when the shape of data clusters is not limited to be spherical. Figure 1 shows an example in which the popular *k*-means clustering algorithm can not achieve satisfactory result.



(a) (b)

Figure 1. Clustering results on a dataset with 2 clusters
(a) with *k*-means (b) with *FCC*

To deal with more complex situations, it should be more suitable to take both local neighborhood and global information into consideration when deciding the membership of each data instance. In this paper, we propose a new clustering algorithm which uses fuzzy connectedness as the similarity metric.

The concept of fuzzy connectedness was first introduced by Rosenfeld in 1979 [8]. It can be utilized to deal with the spatial vagueness when the spatial entities of data do not have homogeneous interiors and sharply boundaries. Fuzzy connectedness has been successfully applied to image processing. In an image segmentation framework introduced by Udupa et al. [10], fuzzy affinities are assigned to the target data object during the classification. The affinity between a pair of pixels is defined as a weighted function of their adjacencies in the coordinate space, the intensity space, and the intensity gradient space.

In a recent work, Herman and Carvalho [3] generalized the concept of fuzzy connectedness. For a finite set of data points, the strength of the link from one point to another is defined as the reciprocal of the distance between them and it can be automatically defined based on statistical properties of the links within regions of points identified as belonging to the same groups. A chain is a sequence of ordered points and the strength of a chain is its weakest link. Taking fuzzy connectedness as the similarity metric, the task of a clustering process is to find the strongest path of every data point and assign the point to the group that path leads to.

Basic definitions that we use in this paper are presented below.

Definition 1: For a positive integer *K*, a *K*-FCC of a set *D* (of points) is a function which maps each point $c \in D$ into a (*K*+1)-dimensional vector $\sigma^c = (\sigma_0^c, \sigma_1^c, \sigma_2^c, ..., \sigma_K^c)$, such that $\sigma_i^c \in [0,1]$, and for at least one *k*, $1 \le k \le K$, $\sigma_0^c = \sigma_k^c$; for $1 \le m \le K$ *and* $m \ne k$, $\sigma_m^c = 0$ or $\sigma_m^c = \sigma_k^c$.

Definition 2: A fuzzy affinity on *D* is a function: $\varphi : D^2 \to [0,1]$. φ(c, d) is the strength of the link (c,d) .

Definition 3: A chain in $U \subseteq D$ from c^0 to c^N is a sequence $C^{(0,N)} = (c^0, c^1, c^2, \ldots c^N)$ in *U* and the strength of the chain $\varphi_U(C^{(0,N)}) = \min(\varphi_U(c^{n-1}, c^n))$, $1 \le n \le N$.

Given the above definitions, a clustering process is to find the mapping of *K*–FCC which is deduced from the calculation of fuzzy affinities between pairs of points. In this process, the definition of the fuzzy affinity function φ is very essential; one can include the statistical information of each cluster to make the results more reliable. As shown in the example of Figure 1, when only the Euclidean distance between pairs of points is considered, it is not possible to separate the two clusters correctly. However, after we involve the statistical information of each cluster, e.g., the distribution of Euclidean distance among each point and its neighbors, the clustering result turns out to be perfect (Figure 1b).

In the proposed method, in order to initialize the clustering process, we need a little prior knowledge. This is in the form of a few labeled data points in each possible cluster. This requirement, however, is usually easy to satisfy in intrusion detection applications. We discuss the details of our clustering algorithm in the next section.

3. METHODOLOGY

The proposed clustering algorithm starts by assigning the seed points into corresponding clusters. For each unlabeled data point *c*, its fuzzy affinity to each cluster U_k, $1 \le k \le K$ is calculated and its mapping vector $\sigma^c = (\sigma_0^c, \sigma_1^c, \sigma_2^c, ..., \sigma_K^c)$ is updated accordingly. Finally, the data point *c* is assigned to the cluster U_k to which *c* has the highest fuzzy affinity ($\sigma_0^c = \sigma_k^c$).

The details of the algorithm are given in Table 1. This is similar to the segmentation algorithm introduced in [3]. However, unlike image data for segmentation, data to be clustered usually have higher (more than 2 or 3) dimensions and there is no strictly defined spatial adjacency (like 4 or 8-connectinity in image data) among different data points. In order to solve this problem, the concept of neighborhood is defined. The neighborhood of a data point consists of a fixed number of its nearest neighbors in the multi-dimensional space. A point is considered to be fuzzy connected only to those points in its neighborhood. Moreover, as defined later, the formula of fuzzy affinity is also different.

Algorithm 1. Fuzzy-connectedness based Clustering (*FCC*)

```
Input: Data $D$, number of clusters $K$, seed points $S(U_k)$ $1 \le k \le K$;
Output: $K$-FCC matrix $\sigma = \{\sigma_k^c \mid c \in D, 0 \le k \le K\}$ and label matrix label(c) for $c \in D$.
1:  for every $c \in D$
2:        for k = 1: K
3:          $\sigma_k^c = 0$
4:  $Heap = \phi$
5:  for k = 1: K
6:      $U_k = S(U_k) \cup$ {nearest neighbors of $S(U_k)$}
7:        $Heap = Heap \cup U_k$
8:        for every $c \in U_k$  $\sigma_k^c = 1$
9:  $Max\_link = 1$
10: while $Max\_link > 0$
11:     for k = 1: K
12:           while $|U_k| > 0$
13:             $d = U_k(1)$;  $U_k = U_k - d$
14:             $NN$ = {nearest neighbors of $d$}
15:             $C = \{c \in NN \mid \sigma_k^c < \min(Max\_link, \varphi_k(d,c))\ \&\ \sigma_0^c \le Max\_link\}$
16:             while $|C| > 0$
17:                  $c = C(1)$;  $C = C - c$
18:                  $l = min(Max\_link, \varphi_k(d,c))$
19:                  if  $Max\_link = l$ and $\sigma_k^c < Max\_link$  then $U_k = U_k \cup \{c\}$
20:                  if  $\sigma_0^c < 1$  then
21:                       if $\sigma_0^c = 0$  then  $Heap = Heap \cup \{c\}$
22:                       for $n = 1: K$  $\sigma_n^c = 0$
23:                  if $\sigma_0^c \le l$  then  $\sigma_0^c = \sigma_k^c = l$
24:       $Heap = Heap - \{c \mid c \in Heap$ and $\sigma_0^c = Max\_link\}$
25:         $Max\_link = Max(\sigma_0^c)$, $c \in Heap$
26: for  every $c \in D$
27:       $label(c) = k$  with $\sigma_k^c = \sigma_0^c$
```

The algorithm starts with an initialization process (steps 1-9). For every data point $c \in D$, its fuzzy mapping vector is initialized with zeros. As shown later in the algorithm, the values in the mappings vectors are used to decide the memberships of the data points. All the seed points are put into corresponding clusters and certain items in their mapping vectors are set to 1. A heap data structure is used to temporarily contain the unlabelled data points. It is set to empty at the beginning. Before the main loop (10-27) of the algorithm, only the seed points $S(U_k)$ and their nearest neighbors are labeled. Their linkage to their cluster is considered the strongest and the value is set to one. Starting with these points, all the clusters try to expand. The strength of the linkage of each unlabeled point to a cluster is the strength of a chain (path) that connects this point to one seed point in that cluster. After all the fuzzy affinities are calculated, we get the *K*-FCC for the dataset *D*. The label of each data point $c \in D$ is finally decided by its mapping vector σ^c. It is assigned to cluster *k* if σ_k^c is the largest element of the vector.

In the *FCC* algorithm, a priority heap is used for the operations of insertion into (step 21) and removal (step 24) from the priority queue. With the definition of nearest neighborhood of an instance, there are no more than a fixed number of nearest neighbors that need to be processed in step 15. If there are *N* instances in *D* and *n* is used to denote the number of nearest neighbors, the computational complexity of *FCC* is $O(N(\log N + nK))$.

There are two issues that are very important to this algorithm: one is related to the definition of neighborhood (*NN*) of a data instance, the other is how to calculate the fuzzy affinity between each pair of instances.

For the first issue, the neighborhood is defined in Euclidean space. A small number (e.g. 4) of neighbors around the target point with the shortest Euclidean distances are defined to be its nearest neighbors. With an indexing scheme such as an R-tree, it is very easy to find the neighborhood in a short time.

In order to calculate the fuzzy affinity of an instance *c* and another instance $d \in U_k$ (target cluster), besides their spatial adjacency, we also involve the statistical information about the cluster U_k. The distribution information of a certain variable (e.g., Euclidean distance between nearest neighbors) of U_k is stored and kept up to date. When a new instance comes, its probability of falling into U_k is calculated using the distribution information and the Euclidean distance between the new instance and its nearest neighbor *d* in U_k.

Let *E(c, d)* be the Euclidean distance between *c* and *d*, and M_k and S_k be the mean and standard deviation of the Euclidean distance between every pair of instances already in U_k. The fuzzy affinity of (*c*,*d*) is then defined as:

$$\varphi_k(c,d) = \begin{cases} 0 & \text{if } c \notin NN(d) \\ \dfrac{\rho_{M_k,S_k}(E(c,d))}{\rho_{M_k,S_k}(M_k)} & \text{otherwise.} \end{cases} \qquad (1)$$

In the definition above, ρ_{M_k,S_k} is the probability density function of a Gaussian distribution with mean M_k and standard deviation S_k.

4. EXPERIMENTAL RESULTS

In order to evaluate *FCC*, we tested the algorithm on a benchmark dataset, the network traffic data from the KDD Cup 1999 Dataset [4]. Two indicators, *Detection Rate* and *False Alarm Rate*, were used to measure the accuracy of the method. The *Detection Rate* shows the percentage of true intrusions that have been successfully detected. The *False Alarm Rate* is defined as the number of normal instances incorrectly labeled as intrusion divided by the total number of normal instances. A good method should provide a high *Detection Rate* together with a low *False Alarm Rate.*

4.1 Data preparation

The KDD dataset includes a wide variety of intrusions together with normal activities simulated in a military network environment. The simulated attacks fall in one of four major categories: DOS (denial of service), R2L (unauthorized access from remote machine), U2R (unauthorized access to local superuser privilege) and Probing (surveillance and other probing). Each instance in the dataset consists of the extracted features of a connection record. These features are either symbolic or continuous.

In the *FCC* algorithm, the Euclidean distance is chosen as the metric to calculate the distance among different instances (points). Since the Euclidean distance can deal only with continuous types of data, all the symbolic features are removed from the data. At the same time, in order to avoid the dominance of any single (or several) feature(s) in the distance calculation, we normalize the data so that the values of every feature have the same range. In the experiments, the values of each feature are normalized with the minimum and maximum values of that feature so that they fall in the range of [0,1].

$$new\_inst[i] = \frac{old\_inst[i] - \min(feature_i)}{\max(feature_i) - \min(feature_i)} \qquad (2)$$

4.2 Results

In our experiments, we consider a subset of the KDD data, which consists of 2776 instances. Except for a few seed points, the labeling information is not utilized in the experiments. It is, however, used after our experiments are performed, to evaluate the results and calculate the *Detection Rate* and *False Alarm Rate*.

In the first experiment, besides the normal instances, instances of four popular attacks are involved: smurf, ipsweep, neptune and back. For each type, only one seed point is labeled at the beginning, and four nearest neighbors are defined as the nearest neighborhood of each instance. As shown in Table 1, most attacks can be distinguished from the normal activities and the *Detection Rate* is as high as 97.8%. At the same time, approximately 3.6% normal activities are erroneously labeled as attacks. Among the instances labeled as attacks, about 95.8% of them are classified to the correct attack types. The difference between this accuracy and the *Detection Rate* is caused by the existing fuzziness among different intrusion types.

To test the ability of the *FCC* algorithm to detect new variations of attacks, we add instances of 10 other attack types, such as nmap, teardrop and dportsweep. While keeping the seed points the same as before, we repeat the experiment. Assuming that the new types of intrusion tend to have stronger fuzzy connectedness to instances of some existing types, we are able to keep the detection rate nearly the same as in the previous experiments. Nearly all the new attack instances are labeled with one of the four previous intrusion types and this keeps the *Detection Rate* at a high level. The actual *Detection Rate* is 98.1%, which is even better than before. At the same time, the *False Alarm Rate* stays at a rather low level.

Table 1. Experimental results of *FCC*

Experiment	Detection Rate	False Alarm Rate
A	97.8%	1.8%
B	98.1%	1.6%

The results of the second experiment clearly demonstrate the great potential of *FCC* in detecting newly emerging intrusions. It is feasible to apply *FCC* on currently available (unlabeled) data, and the clustering results can be used for future incremental learning. As long as the assumption that variations of intrusion types have similar distributions to some known types holds, instances of new intrusion variations will have larger value of fuzzy connectedness to a certain cluster labeled as intrusion and be put into that cluster. Even though we may not know the actual type of the new variation, a correct intrusion alarm will be reported.

4.3 Parameters

In the *FCC* algorithm, two parameters are involved: (a) the number of seed points and (b) the number of neighbors in the neighborhood. The first parameter is decided using prior knowledge about the data. The minimum value is 1, which means that each cluster has at least one instance at the very beginning. Usually, the more seed points we have, the more information about the cluster will be given, and this will possibly provide more accurate results. However, depending on the distributions of different clusters, a small number of seed points may also give satisfactory results, as shown in Table 1.

The number of neighbors used to define the neighborhood is also important. As discussed earlier, when calculating the fuzzy connectedness of an instance to other instances in a certain cluster, both the local information about the distance between two instances and the statistical information about that cluster are involved. An instance *x* is fuzzy connected to another instance *y* only if *y* falls in the neighborhood of *x*. If the number of neighbors is too small, only a few instances tightly close to *x* are considered reachable from *x*, and this may separate *x* from the cluster it really belongs to; when this number is too large, the importance of local connectivity will be reduced and it is possible for two instances far from each other to be considered as fuzzy connected. This may bring two or more clusters close to each other forming a single cluster, which is also not desirable. So, there should be a trade off among the effects of varying the two parameters. Experimental results with different settings of the parameters are displayed in Table 2.

Table 2. Experimental results with different parameters (a) *Detection Rate* (b) *False Alarm Rate*

(a)

		number of seed points		
		1	3	10
number	2	96.3%	93.8%	94.0%
of nearest	4	97.8%	96.7%	97.1%
neighbors	8	98.7%	96.7%	98.0%

(b)

		number of seed points		
		1	3	10
number	2	2.3%	3.2%	4.2%
of nearest	4	1.8%	2.2%	3.3%
neighbors	8	1.5%	2.2%	1.6%

With different configurations of the parameters, while *Detection Rate* stays high and *False Alarm Rate* stays low, there are some changes in the numbers. When the number of nearest neighbors increases from 2 to 4, both *Detection Rate* and *False Alarm Rate* are slightly improved. The reason is that there are more choices for an instance to get connected and this helps it to find the correct cluster. Consequently, fewer attacks will be mislabeled as normal and fewer normal instances will be treated as intrusion. However, there is not always improvement with the increase of the number of nearest neighbors considered. As shown in Table 2, with 3 seed points for each cluster, both *Detection Rate* and *False Alarm Rate* remain the same when the number of neighbors increases from 4 to 8.

Comparing with the other two clustering algorithms introduced in [2, 7], the *Detection Rate* of *FCC* is at least 5% higher. Even though the *False Alarm Rate* with *FCC* is also a little higher, these results are very encouraging since higher *Detection Rate* is more important in most applications. As for the 4 centroid-based clustering approaches discussed in [11], even though the evaluation dataset is different, none of them is reported to have a better result than *FCC*.

An interesting observation from Table 2 is the decrease of performance with the increase of the number of seed points. Since all the seed points are chosen randomly, they may not reflect the actual distributions of those clusters. If some of the seed points are selected from thin areas of the clusters, they may lead to deviation when the clusters expand. Correspondingly, there will be increased errors in the clustering procedure. However, there is no theoretical proof that more seed points will lead to poorer performance. The selection of seed points requires further study.

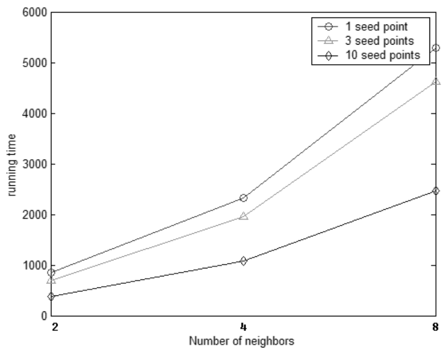


Figure 2. Running time of experiments

The parameters affect not only the accuracy of the *FCC* algorithm, but also its efficiency. Figure 2 shows that the more seed points we have, the less running time the algorithm requires, which means higher efficiency. With a fixed number of seed points, larger number of neighbors leads to reduced efficiency.

5. CONCLUSIONS

In this paper, we introduce a new clustering algorithm, *FCC*, for intrusion detection. This algorithm uses the concept of fuzzy connectedness to calculate the similarities among different data instances. Starting with a single or a few seed points in each cluster, all the data points are dynamically assigned to the cluster that has the highest fuzzy connectedness value (strongest connection). In calculating the fuzzy connectedness, both local distance information and statistical properties of clusters are considered. Comparing with the frequently utilized Euclidean distance, the new similarity metric is more robust, and there is no restriction on the shape of clusters that can be discovered. *FCC* has also some other advantages over the other previously proposed clustering algorithms: no predefined "cluster width", no threshold for "confidence area", etc. Although the *FCC* algorithm requires a parameter K which denotes the number of clusters, most of the time it is not necessary to know an exact value for it. In many cases where we just care about whether an instance is an intrusion and do not need to know about the exact intrusion type, the parameter K can be fixed as a constant, i.e., K=2. With an efficient heuristic algorithm, the time complexity of the clustering process is $O(N\log N+nK)$, where N is the number of data points and n, K are constants. Moreover, this unsupervised learning method can detect not only the known intrusion types, but also their variants. The two other parameters in *FCC*, the number of seed points and the number of neighbors, do not seem to affect greatly the performance of the algorithm. Experimental results on a subset of KDD-99 dataset showed the stability of efficiency and accuracy of the *FCC* algorithm. With different settings, the *Detection Rate* stayed always above 94% while the *False Alarm Rate was* below 4%.

For future work, we need to find out if there is a general accepted definition of fuzzy affinity. With different distribution variable involved in the calculation, the values of fuzzy affinity may be different and it may lead to different clustering results. Even though the one we employ in equation (1) gives good results on the KDD-99 data set, it is not necessary that it will be appropriate for other datasets.

ACKNOWLEGEMENTS

The authors are grateful to G. Herman and B. Carvalho for providing helpful comments and source code for a fuzzy connectedness algorithm for image segmentation. This material is based upon work supported by NSF under Grant No. IIS-0237921.

References

[1] R. Bace and P. Mell, "Intrusion Detection Systems", *NIST Special Publications SP 800-31*, November, 2001.
[2] Y. Guan, A. Ghorbani and N. Belacel. "Y-means: A Clustering Method for Intrusion Detection". Proceedings of Canadian Conference on Electrical and Computer Engineering. Montreal, Quebec, Canada. May 4-7, 2003.
[3] G. Herman and B. Carvalho. "Multiseeded Segmentation Using Fuzzy Connectedness". IEEE Transactions on Pattern Analysis and Machine Intelligence 23(5): 460-474, 2001.
[4] KDD Cup 1999 data. University of California, Irvine.
http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html,1999.
[5] X. Li and N. Ye, "Decision tree classifiers for computer intrusion detection", *Real-time system security*, Nova Science Publishers, 2003.
[6]M. Phung, Data Mining in Intrusion Detection? Intrusion Detection FAQ. October 24, 2000. http://www.sans.org/newlook/resources/IDFAQ/data_mining.htm
[7] L. Portnoy, E. Eskin and S. Stolfo. ``Intrusion detection with unlabeled data using clustering''. Proceedings of ACM CSS Workshop on Data Mining Applied to Security (DMSA-2001). Philadelphia, PA: November 5-8, 2001.
[8] A. Rosenfeld, "Fuzzy Digital Topology," Information and Control, vol. 40, pp. 76-87, 1979.
[9] J. Ryan and M. Lin, "Intrusion Detection with Neural Networks", *Advances in Neural Information Processing Systems 10*, MIT Press, June, 1998.
[10] J. Udupa and S. Samarasekera. Fuzzy Connectedness and Object Definition: Theory, Algorithms, and Applications in Image Segmentation. *Graphical Models and Image Processing*, 58(3):246–261, 1996.
[11] S. Zhong, T. Khoshgoftaar and N. Seliya, "Evaluating Clustering Techniques for Network Intrusion Detection", Proceeding of 10th ISSAT Int. Conf. on Reliability and Quality Design, pp. 149-155. Las Vegas, Nevada, USA. August 2004.